321-4002 – Software Engineering

Team Project for One or Two Member Teams

Delivery Deadline: 15/01/2025

GOALS

The main goal of this software project is to enable students to participate and receive relative experience in the cooperative development of software systems. A special emphasis is put on all the activities of the software development process and especially the requirements engineering, design, implementation and testing ones. In addition, an important attention is given to the use of particular technologies that promote cooperation, construction automation, dependency management and unit testing, such as Maven, JUnit (if Java is used) & Git. Finally, a special focus is made on the development of service-oriented systems, which include one or more RESTful services that can be developed through the assistance of a respective framework, like Jersey or Spring Boot.

Please note that some of the mentioned technologies (Maven, JUnit, Jersey) are language-specific. So, in case you decide to use a different implementation language, you need to adopt similar technologies (e.g., pip/Poetry/Pipenv, Django REST Framework, pytest for Python).

PROJECT TOPIC

FUNCTIONS / REQUIREMENTS

You need to develop the backend of a music festival management system through implementing relevant RESTful web services, which will interconnect with an underlying database for the proper management of all the festival-related entities. The main functionality to be supported by this backend will concern the management of music festivals and of performances (i.e., musical acts) submitted to these festivals. Please note that the management of a festival goes until the final scheduling and announcement of its lineup and does not include further activities (like ticket sales or attendee registration). Thus, the system should handle the entire process from festival creation, through performance submissions, reviews, and scheduling, up to the final lineup announcement. Performance management must include the following functions:

- *Performance creation:* Creation of a new performance. The performance should belong to a specific festival. For accepting this performance creation, the performance name should be unique within the festival, and all obligatory information about the performance should be supplied. The name of the requestor (performance creator) must be included as the main artist of the performance. The performance id and creation date must be automatically generated by the backend, and the requestor must be given the user role of ARTIST for the specific festival.
- *Performance update:* The information about a performance can be updated. This can include information, such as the name, description, genre, band members, duration, technical requirements, merchandise items, setlist and preferred time slots (of rehearsal and performance). Only ARTISTS can update their own performance.
- *Band member addition:* The performance's main artist can indicate whether another user of the system can manage the performance. This user's name should be included in the performance's band members for this addition to be accepted. Once the addition is accepted, the added user must be given the user role of ARTIST for the specific festival.
- *Performance submission:* Once a performance's information has been completed, it can be formally submitted to the festival. Upon successful submission, the performance's status changes from CREATED to SUBMITTED. The submission can be acceptable only if the festival

has a suitable state (i.e., the SUBMISSION one), allowing such a submission, and the performance details are complete. The latter means that the following fields have non-empty values: name, description, genre, duration, band members, technical requirements, setlist, merchandise items, preferred rehearsal times and preferred performance time slots.

- *Performance withdrawal*: The performance can be withdrawn from the festival by its ARTIST(s) only before the performance reaches the SUBMITTED state. When withdrawn, the performance is completely deleted from the system as it has not entered the formal review process yet.
- Staff assignment: During the ASSIGNMENT state, only one STAFF member must be assigned to each performance as the primary performance's handler. The assignment is accepted as long as the staff member is registered as STAFF for the festival. The assigned STAFF member will be responsible for checking and coordinating with the respective ARTISTs the technical requirements and logistics of the performance, while he/she will be the exclusive reviewer of that performance.
- Performance review: The assigned STAFF member, who is already familiar with the performance details and has direct contact with the performance group, must provide a comprehensive review of the performance's suitability for the festival. The review must contain both a numerical score and detailed written comments that reflect the performance's technical and artistic evaluation. The review is accepted only if the festival is in the REVIEW state. This approach ensures that the evaluation comes from someone with deep understanding of both the technical requirements and the performance's overall fit with the festival.
- *Performance approval*: The performance is approved but has not yet been finally scheduled in the festival. It may need modifications based on reviewer comments. This approval is allowed only if the festival is in the SCHEDULING state.
- *Performance rejection*: A performance can be rejected from the festival in two alternative ways:
    1. Manual rejection by ORGANIZERS:
        - During SCHEDULING state: Based on the review comments, if the performance is deemed unsuitable for the festival
        - During DECISION state: If the performance's final submission does not adequately address the reviewer comments
    2. Automatic rejection by the system:
        - During DECISION state: Any approved performance that has not been finally submitted is automatically rejected by the system

  In all cases, the reason for rejection must be recorded. For manual rejections, ORGANIZERS must provide the rejection reason. For automatic rejections, the system records "Failed to submit final performance details" as the reason. This rejection process ensures that ORGANIZERS only need to review performances that have properly completed all required steps.

- *Performance final submission:* The performance's details might need to change to address reviewer comments. These changes, including the proposed setlist, time slots for the performance and rehearsal times, are submitted together as a single operation for the final submission. The setlist includes the planned songs or pieces to be performed, while rehearsal and performance times are coordinated between STAFF (members) and ARTISTS. Once this final submission is done, the approved performance takes its final form and might be scheduled in the festival. This final submission is allowed only if the festival is in the FINAL_SUBMISSION state. After a successful final submission, no further modifications of any kind will be allowed to the performance details. Note that regular performance updates (not related to reviewer comments) are only allowed until the performance is formally submitted (during the SUBMISSION state).
- *Performance acceptance*: The performance has been finally accepted and scheduled in the

festival. This acceptance is allowed only if the festival is in the DECISION state.

- *Performance search*: It is allowed to search for a performance based on its name, artists, and genre. If more than one word is given for a field, then all the words need to be included in this field for a performance to be matched. If no word is given for any of the aforementioned three criteria, then all the festival performances are matched. A set of performances must be returned that satisfy the search criteria. Based on the user role, this set can be further filtered before it is finally returned. The performance set must be sorted first by genre, and within each genre, by performance name.
- *Performance view*: Presentation of the performance information based on its identifier. Depending on the user role, some fields of the performance might be discarded from the presentation.

On the other hand, the management of a festival includes the following functions:

- *Festival creation:* The festival is created as long as its name is unique and all of its obligatory information has been supplied. The festival id and creation date must be automatically generated by the system in this case. The system should also assign the CREATED state to the generated festival. Further, the creator of the festival must be given the user role of ORGANIZER for this festival.
- *Festival update:* The information about a festival can be modified. This includes the name, description, dates, venue layout including stages, vendor areas and facilities, financial information such as budget tracking, costs for performances, logistics and expected revenue, vendor management including food stalls and merchandise booths, as well as the staff and organizers. However, the creator of the festival cannot be removed from the Organizers (i.e., its role for this festival cannot be removed). All venue, budget, and vendor related updates must be completed before the festival reaches the ANNOUNCED state.
- *Organizers addition:* This includes the addition of one or more registered users as Organizers of the festival. In case a user is already an Organizer, his/her addition is ignored. All the added users must be given the role of ORGANIZER for the festival.
- *Staff addition:* This includes the addition of one or more registered users as Staff members of the festival. In case a user is already a Staff member, his/her addition is ignored. All the added users must be given the role of STAFF for the festival.
- *Festival search:* The user should be able to search for festivals. The search criteria can include the name of the festival, its description, dates, and venue. If more than one word is given for a field/criterion, then all the words need to be included in this field for a festival to be matched. If no word is given for any of the aforementioned criteria, then all the festivals are matched. A set of festivals must be returned that satisfy the search criteria (if the latter are given). Based on the user role (see roles), this set can be further filtered before it is finally returned. The festival set to be returned must be sorted based on the festival date followed by festival name.
- *Festival view:* Depending on the user role, some fields of the festival might be discarded from the presentation.
- *Festival deletion:* A festival can be deleted by any of its Organizers only if it is in its initial state (CREATED).
- *Submission start:* The submission of performances is now allowed for the festival. The state of the festival transits from CREATED to SUBMISSION.
- *Stage manager assignment start:* This action is permitted only if the current festival state is SUBMISSION. After executing this action, the state of the festival becomes ASSIGNMENT and the assignment of staff members to the submitted performances can start.
- *Review start:* This action is permitted only if the current festival state is ASSIGNMENT. After executing this action, the state of the festival becomes REVIEW and the review of the submitted performances is allowed.
- *Schedule making:* This action is permitted only if the current festival state is REVIEW. After executing this action, the state of the festival becomes SCHEDULING and the approval or rejection of the submitted performances is allowed, as well as their tentative scheduling.

- *Final submission start*: This action is permitted only if the current festival state is SCHEDULING. After executing this action, the state of the festival becomes FINAL_SUBMISSION. During this state, artists can submit final versions of their approved performances, including any required changes based on reviews.
- *Decision making*: This action is permitted only if the current festival state is FINAL_SUBMISSION. After executing this action, the state of the festival becomes DECISION. During this state, final decisions (accept or reject) are made on the submitted performances. Any approved performances that were not finally submitted by this point automatically become REJECTED.
- *Festival announcement*: This action is permitted only if the current festival state is DECISION. After executing this action, the state of the festival becomes ANNOUNCED. This is the final state where the festival content is locked and can be publicly announced.

Most of the system functions must be rapidly executed (at most 5-10 seconds of execution time per request/call). The system must also be reliable. This means that it should not exhibit internal errors while it needs to produce suitable error messages in case it is wrongly utilized by its users (e.g., wrong or invalid or incomplete input is supplied by the user in their request). This also means that the system must reliably avoid performing multiple times the same successfully executed function (e.g., double creation of a performance) unless the function is idempotent (i.e., it always produces the same result as, e.g., in the case of viewing a performance). Further, the system must be secure. This means that its users must be authenticated and mapped to suitable roles while the access to its functions must be subject to suitable access (control) rules. Additionally, the system should implement rate limiting to prevent abuse, especially for functions like performance submissions or festival searches. Finally, the system must support transactionality, especially with respect to the data that are stored in the underlying database. In other words, there must be consistency in the way the database is updated while this update should be successful only if the corresponding system function (e.g., festival update or performance scheduling) has been completely and properly executed. The system should also implement proper error handling and logging mechanisms to facilitate troubleshooting and maintain an audit trail of all significant actions, which is crucial for managing large-scale events like music festivals.

ROLES

There are 5 main user roles that need to be supported by the system:

- *VISITOR*: An anonymous (i.e., non-authenticated) user. Visitors can only search and view festivals and performances. They are not allowed to perform any other management function. The viewing is also restrained: a visitor can only see accepted performances in a festival and only their name, genre, scheduled time, and main artist(s). Further, the visitor can only see basic information about a festival including its name, dates, venue, and description as well as the names of its organizers.
- *ARTIST*: An authenticated user that has either created a performance or is a band member of a created performance. The role is specific to a certain festival. An artist has access to the following functions: performance creation, updating, band member addition, submission, final submission, and withdrawal. In addition, he/she has all the rights of a VISITOR. However, an ARTIST can see most of the information about his/her performances (including full details, status, review scores and comments but not reviewer names), irrespective of their state.
- *ORGANIZER*: An authenticated user that has either created a festival or has been assigned as an organizer of the festival. As such, this is a role specific to a certain festival. An ORGANIZER has all the rights of a VISITOR. However, organizers can see all the details about the festivals for which they are responsible as well as all the details of all the performances in these festivals. Further, they have access to the following performance management functions: performance approval, rejection, and acceptance plus stage manager assignment for the festivals for which they are responsible. Finally, they have access to all management functions for the festivals for which they are an ORGANIZER.

- *STAFF*: An authenticated user that can be assigned to handle one or more performances of a specific festival. As such, this role is festival-specific. A STAFF member, once assigned to specific performances, can participate in their review process. They also have all the rights of a VISITOR. Further, they can see the full details of only those performances to which they are specifically assigned. A STAFF member can only review the performances they are assigned to, preventing any potential conflicts of interest with other performances in the festival.

Please note that all registered users initially do not have a specific role in the system. However, they do have the right to create performances or festivals. As such, they can immediately become ARTISTS or ORGANIZERS. Once a festival is being managed, then there is a possibility that other authenticated users can become STAFF or ARTISTS (for that festival). An authenticated user can play multiple roles across different festivals, but within a single festival, roles must be strictly separated to prevent conflicts of interest. In particular, if a user is an ARTIST for a festival, he/she cannot simultaneously be a STAFF member for that same festival, as this could create situations where he/she might have influence over other artists' performance reviews or scheduling decisions. An authenticated user that is an ORGANIZER for a festival cannot be an ARTIST or STAFF member for that same festival. Each role should have clearly defined responsibilities and access limitations to maintain the integrity of the festival management process and prevent any potential sabotage or unfair advantage.

The following table summarizes the access rights of the four aforementioned roles:

Role	**Allowed Functions**	**Comments**
VISITOR	View/search festivals and performances	Only scheduled performances can be viewed. The amount of information that can be viewed for festivals and performances is restrained
ARTIST	All allowed functions for VISITOR + performance creation, updating, submission, final submission & withdrawal as well as band member addition	Can view all details about own performances or performances for which they are a band member
ORGANIZER	All allowed functions for VISITOR + all festival management functions + some performance management functions: performance approval, rejection, and acceptance plus stage manager assignment	Can view all details about the festivals for which they are responsible and about the performances in these festivals
STAFF	All allowed functions for VISITOR + performance review	Can view also the full details of performances they are assigned to manage

ENTITIES

The system should be in the position to manage the following 3 main entities:

- *Performance*: is characterized by a single identifier, which needs to be automatically produced by the system during its creation. The same holds for the performance's creation date, which also needs automatic production. A performance has a unique name, a description, genre, duration (set length), and band members. As a performance maps to users of the system as artists, it needs to be correlated with these users. The name, description, genre, duration, band members, and user correlation are considered as obligatory information. The technical requirements (such as equipment requirement, stage setup, and sound/lighting preferences), setlist (i.e., planned songs or pieces to be performed), merchandise items (including details such as name, description, type of item, and price), preferred rehearsal times and preferred performance time slots are optional but need to be non-empty when the performance is submitted or finally submitted. The technical requirements can be a PDF or TXT file. Finally, a performance requires to have one user assigned to it (with role STAFF) as its stage manager. This assignment is conducted at a specific festival state by calling a specific performance management function so only then this association should be supplied (thus, it must be initially empty). Similarly, the reviewer comments and score can only be supplied during the performance's review. A performance must have a specific state at any time point. The possible states are the following: (a) CREATED, (b) SUBMITTED, (c) REVIEWED, (d) REJECTED, (e) APPROVED and (f) SCHEDULED. Initially, the performance state is CREATED. If the performance gets successfully submitted, it moves into the SUBMITTED state. If the performance is reviewed (in the SUBMITTED state), it transits to the REVIEWED state. If the performance is rejected (while in the REVIEWED or ANNOUNCED state), it enters the REJECTED state, which is final. If the performance is approved (while in the REVIEWED state), it transits into the APPROVED state. Once being in APPROVED state, the performance can move into the SCHEDULED or REJECTED states depending on whether it is scheduled/accepted or rejected, respectively. The SCHEDULED and REJECTED states are final states.
- *Festival*: is characterized by a single identifier, which needs to be automatically produced by the system during its creation. The same holds for the festival's creation date, which also needs automatic production. A festival has a unique name, description, dates and venue, which are considered as obligatory information, along with at least one ORGANIZER (an authenticated user) that must be assigned at creation time. In addition, venue layout (stages, vendor areas and facilities), budget (budget tracking, costs for performances, logistics and expected revenue) and vendor management (food stalls and merchandise booths) are optional but have to be completed before the festival reaches the ANNOUNCED state. A festival needs to be associated with its performances and can be associated with multiple ORGANIZERS and STAFF members, if needed. Finally, a festival should be associated with one or more STAFF members (i.e., authenticated users). The STAFF members set should be finalized before the stage manager assignment activity. Thus, after the SUBMISSION state of the festival, no changes to the STAFF members set is allowed. A festival must have a specific state at any time point. The possible states are the following: (a) CREATED, (b) SUBMISSION, (c) ASSIGNMENT, (d) REVIEW, (e) SCHEDULING, (f) FINAL_SUBMISSION (g) DECISION and (h) ANNOUNCED. Initially, a newly created festival is at the CREATED state. Then, it can transit to the ANNOUNCED state, i.e., the final one, sequentially in a manual manner through the execution of respective festival management functions. The festival moves from the CREATED to the SUBMISSION state by executing the submission start function. Once this is done, performances can be submitted but not reviewed. The festival moves from SUBMISSION to ASSIGNMENT state by executing the stage manager assignment start activity. This indicates that performances cannot be submitted anymore and only staff assignments are allowed. The festival transits from ASSIGNMENT to REVIEW state by executing the review start function. This signifies that staff assignment has been finished and performance reviewing can proceed. The festival moves from REVIEW to SCHEDULING state by executing the schedule making function. This indicates that no review is then possible and only scheduling of performances is allowed (i.e., approval or rejection of

performances). The festival moves from SCHEDULING to FINAL_SUBMISSION state by executing the final submission start function. During this state, artists can submit final versions of their approved performances, including any required changes based on reviews. The festival moves from FINAL_SUBMISSION to DECISION state by executing the decision making function. During this state, final decisions (either accept or reject) are made on the submitted performances. Any approved performances that were not finally submitted by this point automatically become REJECTED. Finally, the festival moves from DECISION to ANNOUNCED state by executing the announcement function. No more changes to the festival's lineup or schedule are allowed in this case.

- *User*: must have a username and password based on which their authentication can be performed. Further, they must have a (full) name. Finally, they can have different roles in different festivals (e.g., ARTIST in one festival and ORGANIZER in another one) but at most one role in a specific festival.

ASSUMPTIONS/CONSTRAINTS

To facilitate the project implementation, it can be assumed that:

- There exists a user management system responsible for managing all users, irrespectively of their roles in the system. As such, user authentication can be realised by utilising the shared, underlying database (between the user and festival management systems) where it is checked whether the current user has supplied a correct and valid username and password combination. Both user authentication and authorization are functions that must be supported by the festival management backend. In this way, it is considered that they are not supplied as services by the user management system.

First Project Part (15/100)

Based on the above analysis of the project high-level, user requirements, you must produce a set of detailed system requirements, both functional and non-functional. You are free to make any assumption in case you regard that the above analysis is vague or ambiguous, while producing the needed system requirements. All generated requirements need to be incorporated in the respective part of the report to be delivered. You can use natural or structured natural language to specify the needed requirements.

Second Project Part (30/100)

Based on the extracted system requirements, you should now proceed to system design. In this case, you need to produce and incorporate, at the appropriate place in the report to be delivered, the following diagrams:

- A context diagram for modelling the environment that surrounds the software system to be developed
- Use case diagrams to model the main use cases of the system
- A component diagram to model the system architecture
- A set of activity diagrams in order to detail the expected behaviour in the context of some of the modelled use cases
- A set of sequence diagrams in order to detail the expected behaviour in the context of some of the modelled use cases. Please note that the set of activity and sequence diagrams must jointly cover all the use cases modelled for the system
- An Entity-Relationship (ER) diagram or a database diagram to facilitate the design and implementation of the underlying database

Every incorporated diagram must be accompanied with its short textual analysis / explanation within the report to be delivered.

Third Project Part (50/100)

Based on the system models produced in the second project part, proceed with the collaborative implementation of the system by utilizing any programming language you prefer. However, the use of the Java programming language is recommended due to its respective technologies that have been taught and exercised during the laboratorial part of the course. The main system functions must be supplied in the form of one or more RESTful services. To this end, a suitable framework for developing RESTful services must be adopted (such as Jersey or Spring Boot if the implementation language is Java).

The collaborative implementation of the system must be realised by using Git and a private repository in Github. In the private repository, the contributing members should be the members of the team and the course instructors, who need to be invited (user names: kkritikos & alfaaegean). Apart from the basic system source code, unit testing code has also to be implemented (by utilising a respective framework like JUnit). Finally, the use of Maven is suggested, in case Java is used as an implementation language, due to this ability to automatically manage the code dependencies and to automatically produce the artifacts for the various related software construction activities (e.g., main and test code packages, test reports, etc.). In case of using any other programming language apart from Java, it is recommended that any corresponding framework (with respect to Maven) should be adopted.

The main deliveries for this project part must be the following:

- The repository URL of the implemented system in Github
- Implementation documentation: what is the project structure, a class diagram, analysis of main system classes, instructions on how to compile, package and execute the system code, which are the prerequisite systems/software packages (e.g., web/servlet containers, database servers) that must be already installed in the underlying operating system, SQL scripts for database/table generation.
- Test documentation: on which system parts did the testing focus, which test cases were covered, special preconditions or dependencies that are required for executing the unit tests

All these deliveries must be incorporated on the right place of the report to be delivered.

REPORT (5/100)

Apart from the aforementioned project parts with specific deliveries to be incorporated in the final project report to deliver, this report must also include additional parts, which will be graded with 5/100.

The report structure in terms of its main chapters/units must be the following:

1. *Cover Page*
2. *Summary*: must summarise the main functions and features of the system under development and the respective system environment
3. *Organisation*: you need to stress the following:
    a. Which were the basic roles in the development and which members of the team played what roles
    b. The basic development activities/stages conducted/followed (obviously, they map to the first three project parts) as well as the activities duration and total project development time (in hours)

4. *Requirements* (first project part): you must organise this chapter into two sections, one section for the functional and one for the non-functional requirements. The requirements should be numbered in a hierarchical manner in each section.
5. *Design* (second project part): this chapter must be organized into the following sections:
    a. *System Architecture and Context*: supply and shortly explain the component diagram and context diagram of the system to be developed
    b. *Use Cases*: supply and shortly explain the use cases (diagrams) for the system to be developed
    c. *System Behaviour*: supply and shortly explain activity and sequence diagrams, detailing the user-system interactions and system behaviour in the context of the use cases to be supported
    d. *System Entities*: supply and shortly explain the system's ER / database diagram
6. *Implementation* (third project part): this chapter must be organized into the following sections:
    a. *Github*. Project repository URL in GitHub και contributors. Special repository settings (if applicable). Flow of working in Git and basic development branches during system implementation
    b. Implementation Documentation
    c. Test Documentation
7. *Conclusions*:
    a. The experience attained from this project
    b. The problems and obstacles that were faced
    c. Best practices that were followed or had to be followed, which could be (also) applied in your future (software development) projects

SUBMISSION INSTRUCTIONS

One of the two members (or the sole member) of the team must submit the final report in the relative project of the course's eclass (deadline: until 15/01/2025). The report format can be WORD (.doc(x)) or PDF (.pdf). The name of the report file to be submitted must follow the following pattern: <AM1>_<AM2>_report.<ending> where <AM1>, <AM2> are the matriculation numbers of the two team members, respectively and <ending> is the report file's ending based on its format. In case of a one-member team, the file name pattern is simplified as follows: <AM1>_report.<ending>.

The project will be remotely examined via Zoom. Instructions for this examination will be supplied via a related announcement to be posted in the course's eclass.